

[illegible][illegible][illegible][illegible][illegible][illegible][illegible]

amplified by the R/W AMP 107, the analog signal is converted into a digital signal by the read control unit 110, and the run length limited code is converted into the NRZ code by the ENDEC 106. Next, error detection is performed by the ECC circuit 105, the error is corrected when it occurs in the read data, and the corrected data is stored in the data buffer 111. The data stored in the data buffer 111 is transferred to the host computer 101 through the HDC 104, the interface controller 103 and the interface bus 102.

Recently, PRML (Partial Response Maximum Likelihood) has been used as a next generation signal processing technology, as described in Japanese Unexamined Patent Publication No. 190934/1987, "Viterbi Detection of Class IV Partial Response on a Magnetic Recording Channel", IEEE Transaction on Communications, vol. Com-34, No. 5, May, 1986, pps. 454-461, "Signal Processing System PRML Supports a Large Scale Memory Device of Next Generation", Nikkei Electronics, Jan. 17, 1994 (No. 599), pps. 71-97 and the like. The PRML system detects the most likely data series (bit series of the maximum likelihood) among all the occurable signal series by using the PR (Partial Response) system which performs an effective transfer by allowing inter-code interference of data and by using a decoding method called Viterbi algorithm. There are a number of systems in the PR system depending on what kind of inter-code interference is provided. For instance, PR (1,0,-1) (=PR4) is a system of providing a characteristic of (1D) (1+D) to a recording and reproducing system. In using the PRML system, there are many cases wherein (0,4,4) GCR (Group Coded Recording), or 8-9 conversion code is employed as the run length limited code, as shown in Japanese Examined Patent Publication No. 6699/1991. The (0,4,4) GCR signifies that the run is not smaller than 0 and not larger than 4, and in which the last numeral 4 signifies that the maximum value of the run is 4 in view of every other bit of a data series after encoding. Further, the 8-9 conversion Code is one of codes called block code, which signifies that an 8 bits data is converted into a 9 bits data. The block code converts m bits of an original data series into data having n bits ( $m \leq n$ ). It maps combinations suitable for the recording and reproducing characteristic in the n bits data from all the combinations of the m bits data.

With the larger capacity and higher density of a magnetic disk, the S/N is deteriorated, as described in "Design Acknowledging Medium Defect in Small Scale HDD Starts. Importance of Error Correction Using ECC Enhances". Nikkei Electronics, Aug. 5, 1991 (No. 533), pps. 141-146. As a remedy for the deterioration of the S/N, a method has been used wherein redundancy bits of the error correcting code are increased and a strong error correcting code is added. As such an error correcting code, for instance, an error correcting code called BCH Code (Bose-Chaudhuri-Hocquenghem Code) or Reed-Solomon Code has been commercialized. In such an error correcting code, the redundancy bits should be increased in accordance with an increase in the number of error bits to be corrected.

FIG. 27 shows a system in block diagram form wherein an error correcting code is added using the PRML. In FIG. 27, the ECC circuit 105 in the HDC 104 adds an error correcting/detecting code to the write data stored in the data buffer 111, and the ENDEC 106 converts the code having a code form of NRZ of write data which has been inputted from the host computer 101, into the run length limited code that is suitable for recording on the magnetic disk. Further, a convolution encoding called pre-coding is carried out in the signal processing circuit 110 to perform the PRML, by

which a regularity is provided to the data. Thereafter, in the R/W AMP 107, the voltage of the pre-coded write data is amplified, by which the writing is performed on the magnetic disk 109 by the head 108. In the PR system, since the inter-code interference is included in the reproduced waveform, it is necessary to remove the inter-code interference to reproduce the original data. Therefore, normally, the operation of previously providing an inter-code interference which is inverse to that provided in the recording and reproducing system, is called pre-coding. For instance, an encoding which performs the pre-coding of PR (1,0,-1) (=PR4) to data of the NRZ (Non Return to Zero) system is called interleaved NRZI.

Further, in FIG. 27, in reading data, an electric signal which has been read from the head 108, is amplified by the R/W AMP 107, an error is corrected by ML decoding using the regularity provided to the precoding in the signal processing circuit 110, and the signal is digitized and inputted to the ENDEC 106. The run length limited code is converted into the NRZ code data in the ENDEC 106, successively, an error detecting is performed in the ECC circuit 105, and the error is corrected in case wherein the error has occurred in the read data.

Reference is made to FIG. 29 for an explanation of a behavior of propagation of error occurrence in case wherein, for instance, the PRML of Class 4 is employed as the PRML, a 8-8 converting code is employed as a run length limited code. As shown in FIG. 29, in case wherein an error of 1 bit occurs in reading data that has been recorded on a disk medium, when the data is decoded by the signal processing circuit 110 and is inputted to the ENDEC 106, it becomes an error of 2 bits (on the 8-8 converting code shown in FIG. 29) due to the characteristic of PRML. Further, when the 8-8 converting code is converted into the NRZ signal in the ENDEC 106, it is magnified into an error of 2 bytes (on the NRZ code shown in FIG. 29) since it is encoded in the block code. When the length of data is 512 bytes, redundant bits of 48 bits at a minimum is required to correct a continuous 2 bytes in case of the Read-Solomon code.

Further, in the conventional technology, as shown in FIG. 28(d), the error correcting code (ECC) is added only to the write data, and no consideration is given to a case wherein an error occurs in a BYTESYNC region of a synchronization signal, or an identification portion including an area for storing an identification number that is identification information of each sector, or the like. At present, the BYTESYNC is provided with about 1 byte. However, when an error resistance function is to be provided to the NRZ signal in adopting the PRML or the 8-8 converting code, a BYTESYNC region of 5 bytes or more is necessary by adding redundant bits, which very much deteriorates a format efficiency (a ratio of a data capacity as compared with all the memory capacity). Similarly, at present, a CRC (error check code) of 2 bytes is added to the identification portion. However, to provide an error correction capability, a ECC of 4 bytes or more is necessary by adding redundant bits, which very much deteriorates the format efficiency.

Generally, in performing the error correction, the number of the redundant bits of the ECC is necessary to be two times or more of the number of bits to be corrected. The larger the size of error to be corrected, the more it is necessary to increase the redundant bits, which deteriorates the format efficiency.

As stated above, in the conventional technology, writing is performed by converting a data series into the run length limited code after providing the error correcting code thereto

Further, as mentioned above, in the conventional technology, no consideration has been given to the occurrence of an error in the BYTESYNC region, the identification portion or the like, and it is desirable to provide the error correction capability to these regions.

It is an object of the present invention for resolving the above problem to provide a recording and reproducing apparatus, a digital signal processing apparatus and a method of correcting an error capable of correcting an error with a fewer number of redundant bits.

A recording and reproducing apparatus for recording record or user data to be recorded on a record medium by converting the user data into a predetermined record code, according to the present invention is composed of a record encoding unit for converting the user data into encoded user data of the predetermined record code; an error correcting code generating unit for generating error correcting code data for correcting an error with respect to the encoded user data which has been converted by the record encoding unit; a converting unit for converting the error correcting code data which has been generated by the error correcting code generating unit into encoded correcting code data adapted to the record code; a writing unit for writing a write data including the encoded user data which has been converted by the record encoding unit and the encoded error correcting code data which has been converted by the converting unit such that the encoded error correcting code data is adapted to the record code; a reading unit for reading read data including the encoded user data and the encoded error correcting code data which have been written on the record medium; an inverse converting unit for inversely converting the encoded error correcting code data which has been read by the reading unit into the original error correcting code; an error correcting unit for correcting an error with respect to the encoded user data which has been read by the reading unit based on the original error correcting code data which has been inversely converted by the inverse converting unit with respect to the encoded user data; and a decoding unit for decoding the encoded user data having an error corrected by the error correcting unit to the original user data.

The error correcting code generating unit generates the error correcting code data for correcting an error with

FIG. 6 is an explanatory diagram of an error occurrence pattern in an output of signal processing in accordance with the invention.

FIG. 8 is a block diagram of a RLL encoding circuit and a BCH code generating circuit in accordance with the invention.

FIG. 10 is a block diagram of an on-the-fly error correcting circuit in accordance with the invention.

FIG. 12 is a flowchart of the on-the-fly error correcting circuit in accordance with the invention.

FIG. 14 is a more detailed operational sequence block diagram of the on-the-fly error correcting circuit in the example of this invention.

FIG. 16 is an explanatory diagram showing an error occurrence pattern example in accordance with the invention.

FIG. 18 is an autocorrelation diagram of the BYTESYNC pattern in accordance with the invention.

FIG. 20 is a block diagram of an error allowable BYTE-SYNC circuit in accordance with the invention.

FIG. 22 is a flowchart of a writing operation of a format control unit in accordance with the invention.

FIG. 24 is a block diagram of a ECC circuit in accordance with another embodiment of the invention.

FIG. 26 is a system block diagram of a publicly-known example.

FIG. 28 is a diagram showing a record format of a magnetic disk of the conventional system.

## DESCRIPTION OF THE PREFERRED EMBODIMENTS

Referring now to the drawings wherein like reference numerals are utilized to designate like parts, first an explanation will be given of a system of applying a recording and reproducing apparatus according to the present invention to a magnetic disk device as a first example. In this example, in a magnetic disk device using the run length limited code complying with the characteristic of a magnetic disk, an encoder/decoder of an error correcting code is disposed

between an encoder/decoder of the run length limited code and a disk type record medium, and error correction using error correcting code data is performed on the run length limited code. To achieve this operation, redundant bits of the generated error correcting code data satisfies the run length limitation by periodically inserting "1" or "0" to the redundant bits of the error correcting code data which has been generated by an error correcting code generating circuit.

FIGS. 2(a) and 2(b) are flowcharts, wherein in writing data as shown in FIG. 2(a), the operation performs encoding in which a write data is converted into a run length limited code (S2811). Next, the operation generates an error correcting code (ECC code) (S2812). Then, the operation inserts "1", "0" to satisfy the run length limitation with respect to the error correcting code (S2813). The operation performs a pre-coding processing with respect to the write data and the error correcting code (S2814). Further, the operation writes data on a disk medium (S2815). In reading data as shown in FIG. 2(b), firstly, the operation reads data from the disk medium (S2820). The operation performs a signal processing of decoding (S2821). The operation deletes "1", "0" which have been inserted in writing, with respect to the error correcting code (S2822). The operation performs the error correction using the error correcting code (S2823). The operation performs decoding from the run length limited code to the NRZ signal (S2824). The operation then transfers the NRZ signal to the host computer.

Further, in this example, another error correcting code is added also to the identification portion to provide an error correction capability. An explanation will be given of a format of the magnetic disk used in the first example referring to FIGS. 3(a)–3(d) which illustrate the format of the magnetic disk utilized. As shown in FIG. 3(c), each track is divided into a plurality of sectors. Each sector is provided with an identification portion and a data portion. As shown in FIG. 3(d), the identification portion is divided into regions of PROSYNC, BYTESYNC, C, H, S, F, ECC, and GAP. Further, the data portion is divided into regions of PROSYNC, BYTESYNC, DATA, ECC and GAP. PROSYNC is a region for recording a signal for synchronization, and BYTESYNC is a region for recording a signal for byte synchronization. Notation C signifies a cylinder number, H signifies a head number, S signifies a sector number, F signifies a flag showing whether the sector is effective or not, ECC signifies a region for recording an error correcting code and GAP signifies an ineffective region of the identification portion.

These DATA, ID and the like are written in a run length limited code called an 8–8 converting code. The 8–8 converting code is also called (0,4,4) GCR, which is a code that is constructed such that 0 bit or more and 4 bits or less "0" are always interposed between "1" and "1", as shown in FIG. 5(a). Further, as shown in FIG. 5(b), the (0,4,4) GCR is a code which is constructed such that 4 bits or less of "0" are always inserted between "1" and "1" in the respective one or two data (an odd number series and an even number series) which is generated by dividing the original series of data bit by bit. As mentioned later, in this example, in order not to destruct the series of the odd number series and the even number series by inserting "1", the portion of series of bits after inserting "1" are rearranged.

FIGS. 4(a)–4(c) show the portions of ID, DATA and ECC in more detail. FIG. 4(a) shows the identification portion, and FIGS. 4(b) and 4(c) show the DATA portion. Each of the identification portion and the DATA portion is divided into two bits series of an odd number series and an even number series and different error correcting codes are calculated for

the respective ones of the odd number series and even number series. AS shown in FIG. 4(b), each bits series of the data portion is divided into code languages of 486 bits (54 words) for each series, and an error correcting code of 23 or 22 bits is added to each code language. AS shown in FIG. 4(c), each error correcting code (aE1 through aE18/bE1 through bE18) is arranged to each series similar to the data, and in this example, "1" is inserted at a ratio of once per 4 bits to satisfy the run length limitation of the (0,4,4) GCR. Further, in this example, the order of data of the odd number series "a" and the even number series "b" is rearranged or switched, between before and after the insertion of "1", so that each of the data series of the odd number series "a" and the even number series "b" is processed by the same interleaved series. Further, the bits series of the identification portion is divided into 23 bits and 22 bits. The error in the identification portion can be corrected by adding the error correcting codes of 45 bits as in the data portion.

In this example, a case is adopted as an example in which the generally well-known BCH code is employed as the error correcting code. The BCH code used here, employs a primitive polynomial of 9 degree  $x^9+x^4+1$ , and the generating function employs  $G(x)=(x^9+x^4+1)(X^9+X^6+X^4+X^3+1) = x^{18}+x^{15}+x^{12}+x^{10}+x^8+x^7+x^6+x^3+1$ . The calculation used in the error correction is a modal calculation of 2, and "+" signifies an exclusive OR. Hereinafter, an exclusive OR is shown by "+". The BCH code is a code which can correct a random error of 2 bits or less, and which has  $\alpha$ ,  $\alpha^2$ ,  $\alpha^3$  and  $\alpha^4$  as roots, by putting a root of  $x^9+x^4+1=0$  as  $\alpha$ . Using data as  $D(x)$ , the code language  $C(x)$  is expressed as,

$$C(x) = D(x) \cdot x^{18} + D(x) \bmod G(x) \quad (1),$$

which is constructed as,

$$C(\alpha)-0, C(\alpha^2)-0, C(\alpha^3)-0, C(\alpha^4)-0 \quad (2).$$

FIG. 1 is a block diagram arrangement of a system of the present invention operating in the above-described manner. In FIG. 1, numeral 101 designates a host computer such as a personal computer, a workstation or the like, numeral 102 designates an interface bus for connecting the host computer with the magnetic disk device of this example, and numeral 601 designates the magnetic disk device. The magnetic disk device 601 is composed of an interface controller 103 for connecting the interface bus 102 with the magnetic disk device 601, a CPU 602 for controlling the total operation of the magnetic disk device 601, and a HDC 104 for generating a format of the magnetic disk device 601, correcting an error and controlling the access to the disk, a disk 109 for recording and storing data, a head 108 for writing data to the disk 109, and a AMP 107 for converting a very weak voltage read from the disk 109 by the head 108 into a voltage suitable for signal processing and converting a write signal outputted from a signal processing circuit 110 into a voltage suitable for writing by the head 108, a signal processing circuit 110 for performing a PR processing of class 4 on a write signal outputted from the HDC 104 and outputting it to the AMP 107, further performing a maximum likelihood decoding processing on an analog data outputted from the AMP 107, and converting it into a digital data, and a buffer 111 having a function of temporarily storing a data to be written on the disk 109 and a data read from the host computer.

As shown in FIG. 5(b), the signal processing circuit 110 divides data into an even number series and an odd number series and respectively and independently perform the maximum likelihood decoding by an even number series decoder



The operation of the ECC generating circuit 616 will be described with reference to FIG. 8 and FIG. 9, wherein FIG. 9 illustrates timing charts of the RLL encoding circuit. In FIG. 8, when one block of serial data from the parallel/serial converting circuit 801 is inputted from an input data line

901, the BCH code generating circuit 802 sets the switch 904 to the terminal a. The BCH code generating circuit 802 is a circuit wherein the input data is classified into the a series and the b series bit by bit, each of which is divided by the generating function of  $G(x)=x^{18}+x^{15}+x^{12}+x^{10}+x^8+x^7+x^6+x^3+1$  and leaves the remainder to the shift register when the inputting of data is finished, and the remainder is the error correcting code. When the inputting of data is finished, the BCH code generating circuit 802 sets the switch 904 to the terminal b, and outputs the content of the shift register in which the error correcting code of the odd number series is stored, to a signal line 902 and outputs the content of the shift register in which the error correcting code of the even number series is stored, to a signal line 903. Thereby, the error correcting code (aE1 through aE18/bE1 through bE18 shown in FIG. 4(c)) is generated, by which the BCH code can be generated.

At this instance, the RLL encoding circuit 803 operates as shown in FIG. 9. The error correcting code which is outputted via the signal line 902 and the signal line 903, is latched by the latches 1001 and 1002 in accordance with the control signal of the latch timing control circuit 1004, and "1" is inserted when the count value of the counter 1005 is 1. The flag 1006 is used for rearranging the order of data of the odd number series "a" and the even number series "b", and the flag is set/reset when the value of the counter 1005 becomes 1 from 5, such that the flag 1006 is set when the error correcting code of the even number series "b" is outputted prior to the error correcting code of the odd number series "a" after inserting "1". The multiplexer circuit 1003 selects "1" when the count value of the counter 1005 is 1, selects firstly the odd number series "a" which has been latched in the latch 1001 in case wherein the flag 1006 is not set when the count value of the counter 1005 is 2 through 5, and next, selects the even number series "b" which has been latched in the latch 1002, thereby alternately outputting the odd number series and the even number series. When the flag 1006 is set, the multiplexer circuit 1003 selects the even number series "b" which has been latched in the latch 1002, and next, selects the odd number series "a" which has been latched in the latch 1001.

In this way, in the BCH code generating circuit 802 and the RLL encoding circuit 803, the error correcting code is generated and "1" is inserted such that the odd number series and the even number series are alternately outputted so that the series are not destructed, and such that the run length limitation is maintained. The error correcting code is added to the write data by the multiplexer 804 that is outputted to the PRML 110 shown in FIG. 1. Thereby, the error correcting code (aE1 through aE18/bE1 through bE18) of 18 bits is generated for each of the "a" series and "b" series, "1" of 9 bits are inserted there into, and a total of 45 bits are stored in the ECC portion. Further, the error correcting code is generated to each of data in the data portion and the identification portion. The error correcting code in the identification portion is generated and added to the identification portion in formatting the magnetic disk, and is recorded on the magnetic disk after the pre-coding.

The operation of the on-the-fly error correcting circuit 613 for correcting an error in decoding will be described with reference to FIGS. 10, 11, 12, 114 and 15.

FIG. 10 shows a block diagram of the on-the-fly error correcting circuit 613. As shown in FIG. 10, the on-the-fly error correcting circuit 613 is provided with a RLL redundant code recovering circuit 1101 for rearranging an even number series and an odd number series by deleting an inserted "1", a FIFO 1102, a syndrome calculating circuit

1103 for calculating a state of an error which has occurred in a code language, an "0" detecting circuit 1104 for detecting "0", and an error position calculating circuit 1105 for detecting an error position based on a value which has been calculated by the syndrome calculating circuit 1103. Further, the RLL redundant code recovering circuit 1101 is provided with a "1" deleting circuit for deleting the inserted "1" and an "a" series, "b" series rearranging circuit 2302 for rearranging the even number series and the odd number series.

In FIG. 10, the RLL redundant code recovering circuit 1001 deletes "1" which has been inserted at the "1" inserting circuit in encoding, and rearranges the even number series and the odd number series. Thereafter, the syndrome calculating circuit 1103 calculates a state of an error which has occurred in the code language, and the error position calculating circuit 1105 detects an error position based on a value which has been calculated by the syndrome calculating circuit 1103, as mentioned later. Thereafter, the error is corrected based on the data stored in the FIFO 1102 and the error position which has been detected by the error position calculating circuit 1105, and a serial data is converted into a parallel data that is outputted to the 8-8 decoder, in the serial/parallel converting circuit 1131.

FIG. 11 is a circuit block diagram showing the on-the-fly error correcting circuit 613 in more detail. FIG. 12 is a flowchart showing the processing of a sequencer in the on-the-fly error correcting circuit 613. FIG. 13 shows an outline of an operational sequence of the syndrome calculating circuit 1103, the error position calculating circuit 1105 and the "0" detecting circuit 1104 in the on-the-fly error correcting circuit. FIG. 14 shows an operational sequence of the syndrome calculating circuit 1103, the error position calculating circuit 1105 and the "0" detecting circuit 1104 in the on-the-fly error correcting circuit 613. Further, FIG. 15 illustrates operation timing charts of the on-the-fly error correcting circuit 613.

In FIG. 11, the RLL redundant code recovering circuit 1101 is provided with latches 1120, 1125, 1121 and 1122, a counter 1 for counting the number of data, a counter 2 for counting and circulating 1 through 10, and a multiplexer 1123.

As shown in FIG. 15, in the RLL redundant code recovering circuit 1101 of FIG. 11, during the inputting of data other than the error correcting code, the number of data is counted by the counter 1 and the input data is outputted as it is as the output of the multiplexer until 962th bit. Next, during the inputting of the error correcting code and the error correcting code including "1", when the count value of the counter 2 is 1 or 5, "1" which is inputted at every 4 bits interval is deleted and the "a" series data and the "b" series data are rearranged such that the "a" series data is always prior to the "b" series data, by switching the output of the multiplexer 1123. In this way, the inserted "1" is deleted, and the odd number series and the even number series are arranged in turn.

In FIG. 11, the syndrome calculating circuit 1103 performs a general syndrome calculation in the BCH code, as shown in FIG. 13 and 14. Defining code language as  $C(x)$ , calculation units 1106 through 1110 respectively calculate  $S1=C(\alpha)$ ,  $S2=C(\alpha^2)$ ,  $S3=C(\alpha^3)$ ,  $S4=C(\alpha^4)$ ,  $S5=C(\alpha^5)$ . When there is no error in the code language  $C(x)$ ,  $S1=S2=S3=S4=0$ . A "0" detector 1104 checks whether  $S1$  through  $S4$  are 0, and determines that no error occurs when  $S1=S2=S3=S4=0$ , and the data stored in FIFO 1102 is outputted as it is to an output line 115. When at least one of  $S1$ ,  $S2$ ,  $S3$  and  $S4$  is not 0, the error position calculating circuit 1105 is operated to thereby perform the error correction.

## 12

When an error of 1 bit occurs in the code language  $C(x)$ , the following relationship is established.

$$S1 \neq 0, S1 \cdot S3 + S2^2 = 0,$$

$$S3^3 + S1 \cdot S4^2 + S2^2 \cdot S5 + S1 \cdot S3 \cdot S5 = 0$$

When an error of 2 bits occurs in the code language  $C(x)$ , the following relationship is established.

$$S1 \cdot S3 + S2^2 \neq 0,$$

$$S3^3 + S1 \cdot S4^2 + S2^2 \cdot S5 + S1 \cdot S3 \cdot S5 = 0$$

When an error of 3 bits occurs in the code language  $C(x)$ , the following relationship is established.

$$S3^3 + S1 \cdot S4^2 + S2^2 \cdot S5 + S1 \cdot S3 \cdot S5 = 0$$

The error position calculating circuit 1105 performs the correction by using the above property as shown in FIG. 13 and 14. The operational flow of the error position calculating circuit 1105 is shown in FIG. 12.

In FIGS. 13 and 14, the operation reads data to the syndrome calculating circuit 1103 (S2411, 12511). After reading data (S2512), the operation determines whether the syndromes  $S1$  through  $S4$  are 0 or not (S2513), through which the operation determines whether an error occurs or not (S2412). When an error occurs, the operation checks whether an error of 3 bits or more occurs or not (S2413, S2515), whether an error of 2 bits occurs or not (S2414, S2516) and whether an error of 1 bit occurs or not (S2415, S2517), respectively. When an error of 3 bits or more occurs, the operation finishes the error correcting processing as the error can not be corrected (S2421, S2518). When an error of 2 bits occurs, the operation corrects a first 1 bit (S2420) and thereafter corrects the residual bit (S2416, S2524 through S2533). When an error of 1 bit occurs, the operation corrects the bit (S2416, S2519 through S2523).

The error correction method is performed in accordance with a flowchart shown in FIG. 12. In FIG. 12, first, the operation calculates the syndrome of the "a" series ( $F=0$ ,  $m=a$  series), and performs a similar calculation with respect to the "b" series ( $m=b$  series) (S1201). First, the operation sets a switch of a dummy error adding circuit 1130 to the side of "0", and loads a value of the first stage of the syndrome calculating circuits 1106 through 1110, to the first stage of syndrome calculators 1111 through 1115 (S1202). A "0" detector 1116 detects "0" of  $S1$ . A "0" detector 1117 detects "0" of  $S1 \cdot S2 + S2^2$ . A "0" detector 1118 detects "0" of  $S3^3 + S1 \cdot S4^2 + S2^2 \cdot S5 + S1 \cdot S3 \cdot S5$ . When the "0" detectors 1116, 1117 and 1118 detect 0, or when the "0" detector 1118 detects 1, the operation determines that the error can not be corrected (S1203, S1204-1).

Assuming an example in which an error of 2 bits occurs in the data series of  $a$  as shown in FIG. 16. That is, assuming that original data  $a_m$  and  $a_n$  at positions  $i1$  and  $i2$ , are respectively added with an error value "1", and become  $a_m+1$  and  $a_n+1$ . In this case, the syndrome values  $S1$ ,  $S2$ ,  $S3$ ,  $S4$  and  $S5$  are expressed as follows.

---

[Equation 1]

---

$$S1 = a^{i1} + a^{i2}$$

$$S2 = a^{2i1} + a^{2i2}$$


---

[Equation 1]
$S3 = a^{3i1} + a^{3i2}$
$S4 = a^{4i1} + a^{4i2}$
$S5 = a^{5i1} + a^{5i2}$

$$S1 \cdot S3 + S2^2 = 0$$

(the output of the "0" detector 1117)

$$S3^3 + S1 \cdot S4^2 + S2^2 \cdot S5 + S1 \cdot S3 \cdot S5 = 0$$

(the output of the "0" detector 1118)

Next, the operation shifts the shift register by 15 times (S1210). The operation processes the data of the "a" series once per every two times, and therefore, S1 through S5 are multiplied by  $\alpha^8$ ,  $\alpha^{16}$ ,  $\alpha^{24}$ ,  $\alpha^{32}$  and  $\alpha^{40}$ , respectively, and the values of S'1, S'2, S'3, S'4, and S'5 which are inputted to the first stage of the syndrome circulators 1111 through 1115, are as follows.

$$\begin{aligned} S'1 &= a^{i1+8} + a^{i2+8} \\ S'2 &= a^{2(i1+8)} + a^{2(i2+8)} \\ S'3 &= a^{3(i1+8)} + a^{3(i2+8)} \\ S'4 &= a^{4(i1+8)} + a^{4(i2+8)} \\ S'5 &= a^{5(i1+8)} + a^{5(i2+8)} \end{aligned}$$

Further, the operation shifts the register by  $2(503-i2)$  times ( $S1213$ ,  $S1214$ ,  $S1223$ ,  $S1224$  and  $S1225$ ). Then, the values of  $S'1$ ,  $S'2$ ,  $S'3$ ,  $S'4$  and  $S'5$  are as follows.

$$\begin{aligned} S'_1 &= a^{i+8+503-i2} + a^{i2+8+503-i2} = a^{i1+511-i2} + 1 \\ S'_2 &= a^{2(i+8+503-i2)} + a^{2(i2+8+503-i2)} = a^{2(i1+511-i2)} + 1 \\ S'_3 &= a^{3(i+8+503-i2)} + a^{3(i2+8+503-i2)} = a^{3(i1+511-i2)} + 1 \\ S'_4 &= a^{4(i+8+503-i2)} + a^{4(i2+8+503-i2)} = a^{4(i1+511-i2)} + 1 \\ S'_5 &= a^{5(i+8+503-i2)} + a^{5(i2+8+503-i2)} = a^{5(i1+511-i2)} + 1 \end{aligned}$$

At this moment, the operation sets the switch to the side of "1", and using  $S^1=S^1+1$ ,  $S^2=S^2+1$ ,  $S^3=S^3+1$ ,  $S^4=S^4+1$ , and  $S^5=S^5+1$ , the checks the value of the detector A.  $S^2+S^1 \cdot S^3$  and the value of the detector B.  $S^3+S^1 \cdot S^4+S^2 \cdot S^5+S^1 \cdot S^3 \cdot S^5$  (S1212). At this instance, the value of the detector A, or the "0" detector 1117 and the value of the detector B, or the "0" detector 1118 are 0.

Accordingly, the operation corrects the output of the FIFO  $a_m+1$  to  $a_m+1+1=a_m$  and outputs it (S1216). Next, the operation controls the multiplexers 1116 through 1120, and stores S"1, S"2, S"3, S"4 and S"5 to the syndrome circulators 1111 through 1115 (S1217). Therefore, the following values are stored in the syndrome circulators 1111 through 1115.

[Equation 4]

$$\begin{aligned} S^*1 &= a^{11+511-12} \\ S^*2 &= a^{2(11+511-12)} \\ S^*3 &= a^{3(11+511-12)} \\ S^*4 &= a^{4(11+511-12)} \\ S^*5 &= a^{5(11+511-12)} \end{aligned}$$

Since the correction of 1 bit is finished, and the residual error bit is 1 bit, the operation defines as the detector A="0" detector 1116 and the detector B="0" detector 1117 hereinafter and performs the checking similarly. When the operation shifts the register by  $2(i2-il)$ , the values of S'1, S'2, S'3, S'4 and S'5 are as follows.

[Equation 5]

$$\begin{aligned} S'1 &= a^{11+511-12-il+12}=1 \\ S'2 &= a^{2(11+511-12-il+12)}=1 \\ S'3 &= a^{3(11+511-12-il+12)}=1 \\ S'4 &= a^{4(11+511-12-il+12)}=1 \\ S'5 &= a^{5(11+511-12-il+12)}=1 \end{aligned}$$

At this instance, using  $S^*1=S'1+1$ ,  $S^*2=S'2+1$ ,  $S^*3=S'3+1$ ,  $S^*4=S'4+1$  and  $S^*5=S'5+1$ , the operation checks the value of the detector A, or S"1 and the value of the detector B, or  $S^*2^2+S^*1 \cdot S^*3$ . In this case, the value of the detector A or the value of the "0" detector 1116 is 0. Further, the value of the detector B or the value of the "0" detector 1117 is 0 (S1212). Accordingly, the operation corrects the output of the FIFO  $a_m+1$  to  $a_m+1+1=a_m$  (S1216). Next, the operation controls the multiplexers 1116 through 1120 and stores S"1, S"2, S"3, S"4 and S"5 to the syndrome circulators 1111 through 1115 (S1217). Therefore, the following values are stored in the syndrome circulators 1111 through 1115.

$$\begin{aligned} S^*1 &= 0 \\ S^*2 &= 0 \\ S^*3 &= 0 \\ S^*4 &= 0 \\ S^*5 &= 0 \end{aligned}$$

The correction is finished at this point. The residual data is outputted to the buffer successively through the FIFO. In this way, the on-the-fly error correcting circuit 613 can correct an error of 2 bits or less with respect to one code language.

As stated above, the error correction can be performed in decoding after deleting the inserted "1".

As shown in FIG. 1, the error allowable byte synchronization circuit which can allow an error to some degree, is provided at the prior stage of the on-the-fly error correcting circuit, so that the synchronization can be performed even when an error occurs at the BYTESYNC portion.

The BYTESYNC is a pattern of 2 words which has previously been calculated by simulation, as shown in FIG. 17. FIG. 18 shows how many bits are in agreement in case wherein a pattern by which the pattern of the BYTESYNC is connected with the PROSYNC having a pattern of







By this operation, in the magnetic disk device of this example, the operation can perform the on-the-fly error correction in a state in which the data is encoded by the 8-8 encoder, by which the data can be read from the magnetic disk. Accordingly, the number of bits to be corrected can be restrained to a small value, and the number of redundant bits to be added can be restrained to a small value, thereby making it possible to improve the format efficiency. Further, by correcting an error using the error correcting code on the run length limited code, the correction in consideration of the error occurrence characteristic of the Partial Response Maximum Likelihood decoding that is performed in the earlier stage, can be performed, whereby an efficient correction can be performed, the number of redundant bits to be added can be rendered small, and the format efficiency can be improved.

Although the BCH code is employed as the error correcting code in the above example, other error codes such as a Read-Solomon code or the like may be employed.

In decoding, as shown in FIG. 25, the on-the-fly error correcting circuit is composed of a RLL redundant code recovering circuit 2701 for deleting the inserted "1" and rearranging the even number series and the odd number series, the FIFO 1102, a RS code syndrome calculating circuit 2703 for calculating a state of an error which has occurred in a code language, a "0" detecting circuit 2704 for detecting "0", and a RS code error position and error value calculating circuit 2705 for detecting an error position and an error value based on a value which has been calculated by the RS code syndrome calculating circuit 2703. Also in this case, the error correction by the RS code syndrome calculating circuit 2703 and the RS code error position and error

Although the 8-8 conversion code is employed as the run length limited code in this example, the (1,7) code or the like may similarly be used.

In accordance with the present invention, when providing the data portion with the error correcting capability of approximately 10 portions, the format efficiency can be increased by approximately 5%.

Further, according to this example, by using the above-mentioned BYTESYNC pattern, the BYTESYNC can be provided with the error resistance function and several bits of error can be allowed.

As explained above, according to the present invention, more redundant bits can be deleted. Accordingly, the format efficiency can be promoted. Further, the correction can be performed in consideration of the occurrence tendency of an error which occurs in the signal processing or the like, and therefore, a very efficient correction can be performed.

The number of bits of error occurrence in correcting and detecting can be reduced by providing the HDC with the 8-9 conversion encoder/decoder. Therefore, the error resistance function of the BYTESYNC, the identification portion or the like can be achieved in a simple way and the number of retrials can be reduced.